A Novel Neural Network Model for Joint POS Tagging and Graph-based Dependency Parsing

Dat Quoc Nguyen, Mark Dras and **Mark Johnson**
Department of Computing
Macquarie University, Australia
dat.nguyen@students.mq.edu.au
{mark.dras, mark.johnson}@mq.edu.au

Abstract

We present a novel neural network model that learns POS tagging and graph-based dependency parsing jointly. Our model uses bidirectional LSTMs to learn feature representations shared for both POS tagging and dependency parsing tasks, thus handling the feature-engineering problem. Our extensive experiments, on 19 languages from the Universal Dependencies project, show that our model outperforms the state-of-the-art neural network-based Stack-propagation model for joint POS tagging and transition-based dependency parsing, resulting in a new state of the art. Our code is open-source and available together with pre-trained models at: https://github.com/datquocnguyen/jPTDP.

Keywords: Neural network, POS tagging, Dependency parsing, Bidirectional LSTM, Universal Dependencies, Multilingual parsing.

1 Introduction

Dependency parsing has become a key research topic in NLP in the last decade, boosted by the success of the CoNLL 2006, 2007 and 2017 shared tasks on multilingual dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007a; Zeman et al., 2017). McDonald and Nivre (2011) identify two types of data-driven methodologies for dependency parsing: graph-based approaches (Eisner, 1996; McDonald et al., 2005; Koo and Collins, 2010) and transition-based approaches (Yamada and Matsumoto, 2003; Nivre, 2003). Most traditional graph- or transition-based parsing approaches manually define a set of core and combined features associated with one-hot representations (McDonald and Pereira, 2006; Nivre et al., 2007b; Bohnet, 2010; Zhang and Nivre, 2011; Martins et al., 2013; Choi and McCallum, 2013). Recent work shows that using deep learning in dependency parsing has obtained state-of-the-art performances. Several authors represent the core features with dense vector embeddings and then feed them as inputs to neural network-based classifiers (Chen and Manning, 2014; Weiss et al., 2015; Pei et al., 2015; Andor et al., 2016). In addition, others propose novel neural architectures for parsing to handle feature-engineering (Dyer et al., 2015; Cheng et al., 2016; Zhang et al., 2016; Wang and Chang, 2016; Kiperwasser and Goldberg, 2016a,b; Dozat and Manning, 2017; Ma and Hovy, 2017; Peng et al., 2017).

Part-of-speech (POS) tags are essential features used in most dependency parsers. In real-world parsing, those dependency parsers rely heavily on the use of automatically predicted POS tags, thus encountering error propagation problems. Li et al. (2011), Straka et al. (2016) and Nguyen et al. (2016) show that parsing accuracies drop by 5+% when utilizing automatic POS tags instead of gold ones. Some attempts have been made to avoid using POS tags during dependency parsing (Dyer et al., 2015; Ballesteros et al., 2015), however, these approaches still additionally use the automatic POS tags to achieve the best accuracy. Alternatively, joint learning both POS tagging and dependency parsing has gained more attention because: i) more accurate POS tags could lead to improved parsing performance and ii) the the syntactic context of a parse tree could help resolve POS

Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 134–142,
Vancouver, Canada, August 3-4, 2017.

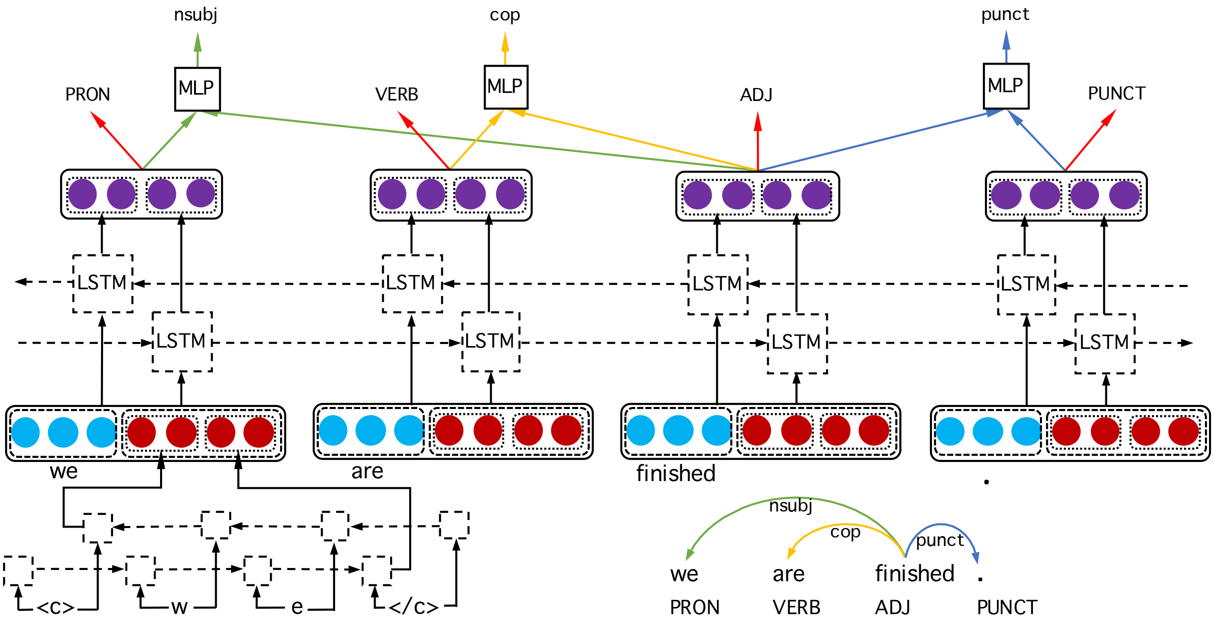

Figure 1: Illustration of our jPTDP for joint POS tagging and graph-based dependency parsing.

ambiguities (Li et al., 2011; Hatori et al., 2011; Lee et al., 2011; Bohnet and Nivre, 2012; Qian and Liu, 2012; Wang and Xue, 2014; Zhang et al., 2015; Alberti et al., 2015; Johannsen et al., 2016; Zhang and Weiss, 2016).

In this paper, we propose a novel neural architecture for joint POS tagging and graph-based dependency parsing. Our model learns latent feature representations shared for both POS tagging and dependency parsing tasks by using BiLSTM—the bidirectional LSTM (Schuster and Paliwal, 1997; Hochreiter and Schmidhuber, 1997). Not using any external resources such as pre-trained word embeddings, experimental results on 19 languages from the Universal Dependencies project show that: our joint model performs better than strong baselines and especially outperforms the neural network-based Stack-propagation model for joint POS tagging and transition-based dependency parsing (Zhang and Weiss, 2016), achieving a new state of the art.

2 Our joint model

In this section, we describe our new model for **j**oint **P**OS **t**agging and **d**ependency **p**arsing, which we call **jPTDP**. Figure 1 illustrates the architecture of our new model. We learn shared latent feature vectors representing word tokens in an input sentence by using BiLSTMs. Then these shared feature vectors are further used to make the prediction of POS tags as well as fed into a multi-layer perceptron with one hidden layer (MLP) to decode dependency arcs and another MLP to predict relation types for labeling the predicted arcs.

BiLSTM-based latent feature representations: Given an input sentence s consisting of n word tokens $w_1, w_2, ..., w_n$, we represent each word w_i in s by an embedding $\mathbf{e}^{(\bullet)}_{w_i}$. Plank et al. (2016) and Ballesteros et al. (2015) show that character-based representations of words help improve POS tagging and dependency parsing performances. So, we also use a sequence BiLSTM (BiLSTM$_{\text{seq}}$) to compute a character-based vector representation for each word w_i in s. For a word type w consisting of k characters $w = c_1c_2...c_k$, the input to the sequence BiLSTM consists of k character embeddings $\mathbf{c}_{1:k}$ in which each embedding vector $\mathbf{c}_j$ represents the j^{th} character c_j in w; and the output is the character-based embedding $\mathbf{e}^{(*)}_{w}$ of the word type w, computed as:

$$\mathbf{e}^{(*)}_{w} = \text{BiLSTM}_{\text{seq}}(\mathbf{c}_{1:k})$$

For the i^{th} word w_i in the input sentence s, we create an input vector $\mathbf{e}_i$ which is a concatenation ($\circ$) of the corresponding word embedding and character-based embedding vectors:

$$\mathbf{e}_i = \mathbf{e}^{(\bullet)}_{w_i} \circ \mathbf{e}^{(*)}_{w_i}$$

Then, we feed the sequence of input vectors $\mathbf{e}_{1:n}$ with an additional index i corresponding to a context position into another BiLSTM ($\mathrm{BiLSTM}_{\mathrm{ctx}}$), resulting in shared feature vectors $\boldsymbol{v}_i$ representing the i^{th} words w_i in the sentence s:

$$\boldsymbol{v}_i = \mathrm{BiLSTM}_{\mathrm{ctx}}(\mathbf{e}_{1:n}, i)$$

POS tagging: Using shared BiLSTM-based latent feature vector representations, then we follow a common approach to compute the cross-entropy objective loss $\mathcal{L}_{\mathrm{POS}}(\hat{\mathbf{t}}, \mathbf{t})$, in which $\hat{\mathbf{t}}$ and $\mathbf{t}$ are the sequence of predicted POS tags and sequence of gold POS tags of words in the input sentence s, respectively (Goldberg, 2016; Plank et al., 2016).

Arc-factored graph-based parsing: Dependency trees can be formalized as directed graphs. An arc-factored parsing approach learns the scores of the arcs in a graph (Kübler et al., 2009). Then, using an efficient decoding algorithm (in particular, we use the Eisner (1996)'s algorithm), we can find a maximum spanning tree—the highest scoring parse tree—of the graph from those arc scores:

$$\mathrm{score}(s) = \operatorname*{argmax}_{\hat{y} \in \mathcal{Y}(s)} \sum_{(h,m) \in \hat{y}} \mathrm{score}_{\mathrm{arc}}(h, m)$$

where $\mathcal{Y}(s)$ is the set of all possible dependency trees for the input sentence s while $\mathrm{score}_{\mathrm{arc}}(h, m)$ measures the score of the arc between the head h^{th} word and the modifier m^{th} word in s. Following Kiperwasser and Goldberg (2016b), we score an arc by using a MLP with one-node output layer ($\mathrm{MLP}_{\mathrm{arc}}$) on top of the $\mathrm{BiLSTM}_{\mathrm{ctx}}$:

$$\mathrm{score}_{\mathrm{arc}}(h, m) = \mathrm{MLP}_{\mathrm{arc}}(\boldsymbol{v}_h \circ \boldsymbol{v}_m)$$

where $\boldsymbol{v}_h$ and $\boldsymbol{v}_m$ are the shared BiLSTM-based feature vectors representing the h^{th} and m^{th} words in s, respectively. We then compute a margin-based hinge loss $\mathcal{L}_{\mathrm{arc}}$ with loss-augmented inference to maximize the margin between the gold unlabeled parse tree and the highest scoring incorrect tree (Kiperwasser and Goldberg, 2016b).

Dependency relation types are predicted in a similar manner. We use another MLP on top of the $\mathrm{BiLSTM}_{\mathrm{ctx}}$ for predicting relation type of an head-modifier arc. Here, the number of the nodes in the output layer of this MLP ($\mathrm{MLP}_{\mathrm{rel}}$) is the number of relation types. Given an arc (h, m), we compute a corresponding output vector as:

$$\mathbf{v}_{(h,m)} = \mathrm{MLP}_{\mathrm{rel}}(\boldsymbol{v}_h \circ \boldsymbol{v}_m)$$

Then, based on MLP output vectors $\mathbf{v}_{(h,m)}$, we also compute another margin-based hinge loss $\mathcal{L}_{\mathrm{rel}}$ for relation type prediction, using only the gold labeled parse tree.

Joint model training: The final training objective function of our joint model is the sum of the POS tagging loss $\mathcal{L}_{\mathrm{POS}}$, the structure loss $\mathcal{L}_{\mathrm{arc}}$ and the relation labeling loss $\mathcal{L}_{\mathrm{rel}}$. The model parameters, including word embeddings, character embeddings, two BiLSTMs and two MLPs, are learned to minimize the sum of the losses.

Discussion: Prior neural network-based joint models for POS tagging and dependency parsing are feed-forward network- and transition-based approaches (Alberti et al., 2015; Zhang and Weiss, 2016), while our model is a BiLSTM- and graph-based method. Our model can be considered as a two-component mixture of a tagging component and a parsing component. Here, the tagging component can be viewed as a simplified version without the additional auxiliary loss for rare words of the BiLSTM-based POS tagging model proposed by Plank et al. (2016). The parsing component can be viewed as an extension of the graph-based dependency model proposed by Kiperwasser and Goldberg (2016b), where we replace the input POS tag embeddings by the character-based representations of words.

3 Experiments

3.1 Experimental setup

Following Zhang and Weiss (2016) and Plank et al. (2016), we conduct multilingual experiments on 19 languages from the Universal Dependencies (UD) treebanks[1] v1.2 (Nivre et al., 2015), using the universal POS tagset (Petrov et al., 2012) instead of the language specific POS tagset.[2] For dependency parsing, the evaluation metric is the labeled attachment score (LAS). LAS is the percentage of words which are correctly assigned both dependency arc and relation type.

[1] http://universaldependencies.org/

[2] Zhang and Weiss (2016) and Plank et al. (2016) experimented on 19 and 22 languages, respectively. For consistency, we use 19 languages as in Zhang and Weiss (2016).

Method	ar	bg	da	de$^\bullet$	en	es	eu$^\bullet$	fa	fi$^\bullet$	fr	hi	id	it	iw	nl	no	pl$^\bullet$	pt	sl$^\bullet$	AVG
	10.3	12.3	15.6	11.9	9.1	7.3	17.8	8.2	24.4	5.7	4.6	13.8	5.7	10.9	18.8	11.2	23.1	10.0	19.9	12.7
PART-OF-SPEECH TAGGING																				
UDPipe	98.7	97.8	95.8	90.7	94.5	95.0	93.1	96.9	**94.9**	95.9	95.8	**93.6**	97.2	94.8	89.2	97.2	96.0	97.4	95.6	95.3
TnT [⊕]	97.8	96.8	94.3	92.6	92.7	94.6	93.4	96.0	93.6	94.5	94.5	93.2	96.2	93.7	88.5	96.3	95.6	96.3	94.9	94.5
CRF [⊕]	97.6	96.4	93.8	91.4	93.4	94.2	91.6	95.7	90.3	95.1	96.0	93.0	96.4	93.6	90.0	96.2	94.0	96.3	94.8	94.2
BiLSTM-aux	**98.9**	**98.0**	**96.2**	92.6	94.5	95.1	**94.7**	**97.2**	**94.9**	95.8	96.2	93.1	**97.6**	**95.8**	**93.3**	**97.6**	**96.4**	**97.5**	**97.6**	**95.9**
Stack-prop	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	95.4
Our **jPTDP**	98.8	97.4	95.8	**92.7**	**94.7**	**95.9**	93.7	96.8	94.6	**96.0**	**96.4**	93.1	97.5	95.5	91.4	97.4	96.3	**97.5**	97.1	95.7
$\bigtriangledown$-Chars	3.1	2.4	3.9	2.3	1.6	0.8	4.3	0.8	5.4	1.1	0.3	3.7	1.4	1.6	6.6	2.7	4.7	3.1	5.7	2.9
DEPENDENCY PARSING																				
UDPipe	76.0	**84.7**	74.8	71.8	80.2	79.7	69.7	79.7	**76.3**	77.8	87.5	73.9	85.7	77.1	71.3	84.5	79.4	**81.3**	80.2	78.5
B'15 [*]	75.6	83.1	69.6	72.4	77.9	78.5	67.5	74.7	73.2	77.4	85.9	72.3	84.1	73.1	69.5	82.4	78.0	79.9	80.1	76.6
PipelineP_{tag}[*]	73.7	83.6	72.0	73.0	79.3	79.5	63.0	78.0	66.9	78.5	87.8	73.5	84.2	75.4	70.3	83.6	73.4	79.5	79.4	76.6
RBGParser [*]	75.8	83.6	73.9	73.5	79.9	79.6	68.0	78.5	65.4	78.9	87.7	74.2	84.7	77.6	72.4	83.9	75.4	**81.3**	80.7	77.6
Stack-prop	77.0	84.3	73.8	74.2	80.7	80.7	70.1	78.5	74.5	**80.0**	**88.9**	74.1	85.8	77.5	**73.6**	84.7	79.2	80.4	**81.8**	78.9
Our **jPTDP**	**79.0**	83.9	**75.8**	**75.8**	**82.0**	**82.4**	**73.2**	**81.5**	75.0	**80.0**	87.3	**75.7**	**86.4**	**79.2**	66.8	**84.9**	**82.5**	79.3	81.7	**79.6**
$\bigtriangledown$-Chars	3.8	4.1	4.5	3.6	1.4	2.3	12.0	1.1	11.1	0.2	0.3	4.1	1.9	1.9	5.4	2.3	10.6	3.4	9.2	4.4

Table 1: Universal POS tagging accuracies and LAS scores computed on all tokens (including punctuation) on test sets for 19 languages in UD v1.2. The language codes with $^\bullet$ refer to morphologically rich languages. Numbers (in the second top row) right below language codes are out-of-vocabulary rates. **UDPipe** is the trainable pipeline for processing CoNLL-U files (Straka et al., 2016). **TnT** denotes the second order HMM-based TnT tagger (Brants, 2000). **CRF** denotes the Conditional random fields-based tagger, presented in Plank et al. (2014). **BiLSTM-aux** refers to the state-of-the-art (SOTA) BiLSTM-based POS tagging model with an additional auxiliary loss for rare words (Plank et al., 2016). Note that the (old) language code for Hebrew "**iw**" is referred to as "**he**" as in Plank et al. (2016). **[⊕]**: Results are reported in Plank et al. (2016). **Stack-prop** refers to the SOTA Stack-propagation model for joint POS tagging and transition-based dependency parsing (Zhang and Weiss, 2016). $\bigtriangledown$**-Chars** denotes the absolute accuracy decrease of our jPTDP, when the character-based representations of words are not taken into account. **B'15** denotes the character-based stack LSTM model for transition-based dependency parsing (Ballesteros et al., 2015). **Pipeline**P_{tag} refers to a greedy version of the approach proposed by Alberti et al. (2015). **RBGParser** refers to the graph-based dependency parser with tensor decomposition, presented in Lei et al. (2014). **[*]**: Results are reported in Zhang and Weiss (2016).

3.2 Implementation details

Our jPTDP is implemented using DYNET v2.0 (Neubig et al., 2017).[3] We optimize the objective function using Adam (Kingma and Ba, 2014) with default DYNET parameter settings and no mini-batches. We use a fixed random seed, and we do not utilize pre-trained embeddings in any experiment. Following Kiperwasser and Goldberg (2016b) and Plank et al. (2016), we apply a word dropout rate of 0.25 and Gaussian noise with $\sigma = 0.2$. For training, we run for 30 epochs, and evaluate the *mixed accuracy* of correctly assigning POS tag together with dependency arc and relation type on the development set after each training epoch. We perform a minimal grid search of hyper-parameters on English. We find that the highest mixed accuracy on the English development set is when using 64-dimensional character embeddings, 128-dimensional word embeddings, 128-dimensional BiLSTM states, 2 BiLSTM layers and 100 hidden nodes in MLPs with one hidden layer.[4] We then apply those hyper-parameters to all 18 remaining languages.

3.3 Main results

Table 1 compares the POS tagging and dependency parsing results of our model jPTDP with results reported in prior work, using the same experimental setup.

Regarding POS tagging, our joint model jPTDP generally obtains similar POS tagging accuracies to the BiLSTM-aux model (Plank et al.,

[3] `https://github.com/clab/dynet`

[4] On English, carried out on a computer with 2.2 GHz Core i7 processor, jPTDP took 6 hours for training with these hyper-parameters, and then obtained a joint tagging and parsing speed of 700 words/second.

2016). Our model also achieves higher averaged POS tagging accuracy than the joint model Stack-propagation (Zhang and Weiss, 2016). There are slightly higher tagging results obtained by BiLSTM-aux when utilizing pre-trained word embeddings for initialization, as presented in Plank et al. (2016). However, for a fair comparison to both Stack-propagation and our jPTDP, we only compare to the results reported without using the pre-trained word embeddings.

In terms of dependency parsing, in most cases, our model jPTDP outperforms Stack-propagation. It is somewhat unexpected that our model produces about 7% absolute lower LAS score than Stack-propagation on Dutch (**nl**). A possible reason is that the hyper-parameters we selected on English are not optimal for Dutch. Another reason is due to a large number of non-projective trees in Dutch test set ($106/386 \approx 27.5\%$), while we use the Eisner's decoding algorithm, producing only projective trees (Eisner, 1996). Without taking "nl" into account, our averaged LAS score over all remaining languages is 1.1% absolute higher than Stack-propagation's.

One reason for our better LAS is probably because jPTDP uses character-based representations of words, while Stack-propagation uses feature representations for suffixes and prefixes which might not be as useful as character-based representations for capturing unknown words. The last row in Table 1 shows an absolute LAS improvement of 4.4% on average when comparing our jPTDP with its simplified version of not using character-based representations: specifically, morphologically rich languages get an averaged improvement of 9.3 %, vice versa 2.6% for others.[5] So, our jPDTP is particularly good for morphologically rich languages, with 1.7% higher averaged LAS than Stack-propagation over these languages.

4 MQuni at the CoNLL 2017 shared task

Our team MQuni participated with jPTDP in the CoNLL 2017 shared task on multilingual parsing from raw text to universal dependencies (Zeman et al., 2017). Training data are 60+ universal dependency treebanks for 40+ languages from UD v2.0 (Nivre et al., 2017a). We do not use any external resource, and we use a fixed random seed and a fixed set of hyper-parameters as presented in Section 3.2 for all treebanks.[6] For each treebank, we train a joint model for *universal* POS tagging and dependency parsing. We evaluate the mixed accuracy on the development set after each training epoch, and select the model with the highest mixed accuracy. Note that for each "surprise" language where there are only few sample sentences with gold-standard annotation or a "small" treebank whose development set is not available, we simply split its sample or training set into two parts with a ratio 4:1, and then use the larger part for training and the smaller part for development.

For parsing from raw text to universal dependencies, we utilize CoNLL-U test files pre-processed by the baseline UDPipe 1.1 (Straka et al., 2016). These pre-processed CoNLL-U test files are available to all participants who do not want to train their own models for any steps preceding the dependency analysis, including: tokenization, word segmentation, sentence segmentation, POS tagging and morphological analysis. Note that we only employ the tokenization, word and sentence segmentation, and we do not care about the POS tagging and morphological analysis pre-processed by UDPipe 1.1. Recall that we perform universal POS tagging and dependency parsing jointly. In addition, when we encounter an additional parallel test set in a language where multiple training treebanks exist, i.e. a parallel test set marked with language code suffix "_pud" such as "ar_pud", "cs_pud" and "de_pud", we simply use the model trained for its corresponding language code prefix, e.g., "ar", "cs" and "de".

Table 2 presents our official parsing results from the CoNLL 2017 shared task on UD parsing (Zeman et al., 2017). We obtain 1% absolute higher averaged scores than the baseline UDPipe 1.1 (Straka et al., 2016) in both categories: big treebank test sets (denoted as **Big** in Table 2) and parallel test sets (denoted as **PUD** in Table 2). Specifically, we obtain a highest rank at **8th** place for the **PUD** category, showing that our parsing model jPTDP is particularly good when it is applied to a real practical application in out-of-domain data. Unlike the baseline UDPipe 1.1 and others, for each surprise language, we simply

[5] To determine a morphologically rich language, we take as a proxy for morphological richness the number of noun cases $>= 4$, with this value obtained from WALS (http://wals.info/) where available or Wikipedia otherwise.

[6] Except for the biggest treebank UD_Czech (cs) consisting of 68K training sentences, due to a limited computation resource, we used 64-dimensional word embeddings and 32-dimensional character embeddings. Then it took 30 hours to complete training process for UD_Czech.

System	**All** (81)	**Big** (55)	**PUD** (14)	**Sma.** (8)	**Sur.** (4)	**R$_{\text{-S}}$**
UDPipe 1.2	69.52_{8}	74.38_{9}	69.00_{9}	53.75_{9}	35.96_{14}	8
UDPipe 1.1	68.35_{13}	73.04_{17}	68.33_{13}	51.80_{15}	37.07_{11}	15
MQuni	68.05_{14}	74.03_{12}	69.28_{8}	51.58_{17}	14.48_{28}	10

Table 2: Official macro-averaged LAS F1 scores of MQuni and baselines from the CoNLL 2017 shared task on UD parsing (Zeman et al., 2017): `http://universaldependencies.org/conll17/results-las.html`. "**All**" refers to the averaged score over all 81 test sets, which is used as the main metric for ranking participating systems. **Big**: the averaged score over 55/81 test sets whose training treebanks are big and have development data available. **PUD**: the averaged score over 14/81 test sets that are additional parallel ones, produced separately and their domain may be different from their training data. **Sma.**: the averaged score over 8/81 test sets whose training treebanks are small, i.e., they lack development data and some of them have very little training data. **Sur.**: the averaged score over 4/81 remaining test sets for surprise languages. Here the *subscript* denotes the official rank out of 33 participating systems. **R$_{\text{-S}}$** is the system rank where the 4 surprise language test sets are not taken into account.

train a joint model just on the sample data of few sentences with gold-standard annotation provided before the test phase, i.e., we utilize neither external resources nor a cross-lingual technique nor a delexicalized parser. So, it is not surprising that we obtain a very low averaged score over the 4 surprise language test sets. When the 4 surprise language test sets are not taken into account, we obtain a rank in top-10 participating systems.

In fact, it is hard to make a clear comparison between our jPTDP and the parsing models used in other top participating systems. This is because other systems use various external resources and/or better pre-processing modules and/or construct ensemble models for dependency parsing.[7] For example, UDPipe 1.2 only extends the word and sentence segmenters of the baseline UDPipe 1.1. Consequently, UDPipe 1.2 obtains 0.1% absolute higher in the macro-averaged word segmentation score[8] and 0.2% higher in the macro-averaged sentence segmentation score[9] than the baseline UDPipe 1.1, resulting in 1+% better in the macro-averaged LAS F1 score though they use exactly the same parsing model. See Zeman et al. (2017) for an overview of the methods, algorithms, resources and software used for all other participating systems.[10]

It is worth noting that for universal POS tagging, we obtain a highest rank at **4th** place for the **Big** category (i.e., 4th on average over 55 big treebank test sets).[11] In this **Big** category, we also obtain better rank than both UDPipe 1.2 and 1.1.

5 Conclusion

In this paper, we describe our novel model for joint POS tagging and graph-based dependency parsing, using bidirectional LSTM-based feature representations. Experiments on 19 languages from the Universal Dependencies (UD) v1.2 show that our model obtains state-of-the-art results in both POS tagging and dependency parsing.

With our joint model, we participated in the CoNLL 2017 shared task on UD parsing (Zeman et al., 2017). Given that we followed a strict closed setting while other top participating systems did not, we still obtained very competitive results. So, we believe our joint model can serve as a new strong baseline for further models in both POS tagging and dependency parsing tasks.

For future comparison, we provide in Table 3 the POS tagging, UAS and LAS accuracies with respect to gold-standard segmentation on the UD v2.0—CoNLL 2017 shared task test sets (Nivre et al., 2017b). Our code is open-source and available at: `https://github.com/datquocnguyen/jPTDP`.

Acknowledgments

This research was supported by a Google award through the Natural Language Understanding Focused Program, and under the Australian Research Council's *Discovery Projects* funding scheme (project number DP160102156). This research

[7] Combining multiple treebanks available for a language or similar languages to obtain larger training data is also considered as a manner of exploiting external data.

[8] Word segmentation results are available at: `http://universaldependencies.org/conll17/results-words.html`

[9] Sentence segmentation results are available at: `http://universaldependencies.org/conll17/results-sentences.html`

[10] Outlined at: `http://universaldependencies.org/conll17/systems-in-a-nutshell.html`

[11] Universal POS tagging results are available at: `http://universaldependencies.org/conll17/results-upos.html`

ltcode	UPOS	UAS	LAS
ar_pud	79.34	68.78	56.81
ar	95.18	84.16	77.82
bg	97.49	88.53	84.20
bxr	43.21	28.79	14.04
ca	98.10	88.62	85.59
cs_cac	98.53	87.52	83.47
cs_cltt	97.20	79.61	74.84
cs_pud	95.96	85.26	79.83
cs	98.41	88.03	84.35
cu	92.81	81.96	73.22
da	95.80	80.87	76.89
de_pud	85.62	78.34	71.34
de	92.83	80.16	75.66
el	96.18	85.07	81.55
en_lines	94.67	79.21	74.60
en_partut	94.17	81.25	76.56
en_pud	94.74	85.49	81.64
en	94.82	85.29	81.64
es_ancora	98.28	88.48	85.50
es_pud	88.59	87.55	80.28
es	96.32	87.66	84.05
et	87.62	69.44	59.15
eu	93.15	77.86	72.56
fa	96.38	85.98	81.91
fi_ftb	92.63	82.48	76.54
fi_pud	96.15	83.15	79.31
fi	94.95	81.89	77.50
fr_partut	95.34	84.75	80.68
fr_pud	89.85	83.50	78.14
fr_sequoia	97.27	86.00	83.25
fr	96.70	87.69	84.51
ga	88.35	73.43	62.24
gl_treegal	92.83	75.45	68.46
gl	96.86	83.77	80.40
got	94.27	77.78	70.27
grc_proiel	94.73	73.25	67.34
grc	86.97	54.87	47.57
he	95.53	86.65	80.91
hi_pud	85.19	64.54	51.97
hi	96.41	90.68	86.71
hr	96.19	85.46	79.32
hsb	51.13	29.88	17.06
hu	91.81	74.05	66.82
id	93.10	83.41	76.84
it_pud	93.51	89.30	85.58
it	97.62	90.28	87.26
ja_pud	97.08	94.40	93.26
ja	96.56	94.07	92.41
kk	51.11	44.25	22.91
kmr	47.72	31.59	18.79
ko	93.47	79.89	74.75
la_ittb	97.44	78.81	74.65
la_proiel	94.23	71.75	64.78
la	83.26	57.79	44.60
lv	90.27	69.28	61.50
nl_lassysmall	95.82	79.74	75.29
nl	91.15	78.47	71.39
no_bokmaal	97.43	88.25	85.33
no_nynorsk	97.07	86.30	83.12
pl	96.18	88.60	82.70
pt_br	97.64	90.40	88.32
pt_pud	88.41	81.49	75.15
pt	96.58	87.88	84.54
ro	96.72	87.04	81.37
ru_pud	86.26	78.88	70.15
ru_syntagrus	98.11	89.73	87.08
ru	95.31	82.14	77.12
sk	94.48	81.26	75.51
sl_sst	88.84	63.25	55.01
sl	96.87	84.75	81.25
sme	33.12	22.80	8.23
sv_lines	94.73	81.52	76.19
sv_pud	91.60	77.73	72.05
sv	96.05	83.35	78.85
tr_pud	72.60	57.14	35.50
tr	93.42	67.39	59.14
ug	72.49	57.79	39.48
uk	88.09	71.03	61.03
ur	92.96	86.05	79.27
vi	86.78	64.88	55.63
zh	92.36	78.57	72.99

Table 3: Universal POS tagging accuracies (labeled as UPOS), UAS and LAS scores of our jPTDP model with respect to gold-standard segmentation on the UD v2.0—CoNLL 2017 shared task test sets (Nivre et al., 2017b). UAS refers to the unlabeled attachment score. **ltcode** denotes the language treebank code. The 4 surprise language tests are *bxr*, *hsb*, *kmr* and *sme*. The 8 small treebank tests are *fr_partut*, *ga*, *gl_treegal*, *kk*, *la*, *sl_sst*, *ug* and *uk*. The 14 parallel test sets are marked with the language code suffix "_pud". The 55 remaining test sets are for big treebanks.

was also supported by NICTA, funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. The first author was supported by an International Postgraduate Research Scholarship—which is an Australian Government Research Training Program Scholarship—and a NICTA NRPA Top-Up Scholarship.

References

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved Transition-Based Parsing and Tagging with Neural Networks. In *Proceedings of EMNLP*. pages 1354–1359.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. In *Proceedings of ACL*. pages 2442–2452.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs. In *Proceedings of EMNLP*. pages 349–359.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING*. pages 89–97.

Bernd Bohnet and Joakim Nivre. 2012. A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing. In *Proceedings of EMNLP-CoNLL*. pages 1455–1465.

Thorsten Brants. 2000. TnT: A Statistical Part-of-Speech Tagger. In *Proceedings of ANLP*. pages 224–231.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*. pages 149–164.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*. pages 740–750.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional Attention with Agreement for Dependency Parsing. In *Proceedings of EMNLP*. pages 2204–2214.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of ACL*. pages 1052–1062.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of ICLR*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proceedings of ACL-IJCNLP*. pages 334–343.

Jason M. Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of COLING*. pages 340–345.

Yoav Goldberg. 2016. A Primer on Neural Network Models for Natural Language Processing. *Journal of Artificial Intelligence Research* 57:345–420.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental Joint POS Tagging and Dependency Parsing in Chinese. In *Proceedings of IJCNLP*. pages 1216–1224.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Anders Johannsen, Željko Agić, and Anders Søgaard. 2016. Joint part-of-speech and dependency projection from multiple sources. In *Proceedings of ACL (Volume 2: Short Papers)*. pages 561–566.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980.

Eliyahu Kiperwasser and Yoav Goldberg. 2016a. Easy-First Dependency Parsing with Hierarchical Tree LSTMs. *Transactions of ACL* 4:445–461.

Eliyahu Kiperwasser and Yoav Goldberg. 2016b. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of ACL* 4:313–327.

Terry Koo and Michael Collins. 2010. Efficient Third-Order Dependency Parsers. In *Proceedings of ACL*. pages 1–11.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies, Morgan & cLaypool publishers.

John Lee, Jason Naradowsky, and David A. Smith. 2011. A discriminative model for joint morphological disambiguation and dependency parsing. In *Proceedings of ACL-HLT (Volume 1)*. pages 885–894.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-Rank Tensors for Scoring Dependency Structures. In *Proceedings of ACL (Volume 1: Long Papers)*. pages 1381–1391.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint Models for Chinese POS Tagging and Dependency Parsing. In *Proceedings of EMNLP*. pages 1180–1191.

Xuezhe Ma and Eduard H. Hovy. 2017. Neural Probabilistic Model for Non-projective MST Parsing. *CoRR* abs/1701.00874.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the Turbo: Fast Third-Order Non-Projective Turbo Parsers. In *Proceedings of ACL (Volume 2: Short Papers)*. pages 617–622.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*. pages 91–98.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics* 37(1):197–230.

Ryan McDonald and Fernando Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of EACL*. pages 81–88.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980* .

Dat Quoc Nguyen, Mark Dras, and Mark Johnson. 2016. An empirical study for Vietnamese dependency parsing. In *Proceedings of ALTA*. pages 143–149.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT*.

Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. 2017a. Universal Dependencies 2.0. http://hdl.handle.net/11234/1-1983.

Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. 2017b. Universal dependencies 2.0 - CoNLL 2017 shared task development and test data. http://hdl.handle.net/11234/1-2184.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, et al. 2015. Universal Dependencies 1.2. http://hdl.handle.net/11234/1-1548.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007b. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering* 13(2):95–135.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proceedings of ACL-IJCNLP*. pages 313–322.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep Multitask Learning for Semantic Dependency Parsing. In *Proceedings of ACL*.

Slav Petrov, Dipanjan Das, and Ryan T. McDonald. 2012. A Universal Part-of-Speech Tagset. In *Proceedings of LREC*. pages 2089–2096.

Barbara Plank, Dirk Hovy, Ryan McDonald, and Anders Søgaard. 2014. Adapting taggers to Twitter with not-so-distant supervision. In *Proceedings of COLING: Technical Papers*. pages 1783–1792.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of ACL (Volume 2: Short Papers)*. pages 412–418.

Xian Qian and Yang Liu. 2012. Joint Chinese Word Segmentation, POS Tagging and Parsing. In *Proceedings of EMNLP-CoNLL*. pages 501–511.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Milan Straka, Jan Hajic, and Jana Strakov. 2016. UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. In *Proceedings of LREC*.

Wenhui Wang and Baobao Chang. 2016. Graph-based Dependency Parsing with Bidirectional LSTM. In *Proceedings of ACL (Volume 1: Long Papers)*. pages 2306–2315.

Zhiguo Wang and Nianwen Xue. 2014. Joint POS Tagging and Transition-based Constituent Parsing in Chinese with Non-local Features. In *Proceedings of ACL (Volume 1: Long Papers)*. pages 733–742.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of ACL-IJCNLP*. pages 323–333.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings IWPT*.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gökırmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Hěctor Martínez Alonso, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadova, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. 2015. Randomized greedy inference for joint segmentation, pos tagging and dependency parsing. In *Proceedings of NAACL-HLT*.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved Representation Learning for Syntax. In *Proceedings of ACL (Volume 1: Long Papers)*. pages 1557–1566.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings ACL-HLT*. pages 188–193.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic Graph-based Dependency Parsing with Convolutional Neural Network. In *Proceedings of ACL (Volume 1: Long Papers)*. pages 1382–1392.